Journal of Computer and System Sciences **64**, 284–307 (2002)
doi:10.1006/jcss.2001.1801, available online at http://www.idealibrary.com on IDEAL®

Minimality of an Automaton Cascade Decomposition for Learning

T. H. Westerdale

Department of Computer Science, Birkbeck College, University of London, Malet Street, London WC1E 7HX, England
E-mail: tom@dcs.bbk.ac.uk

Received February 19, 1999; revised October 10, 2001

We already know how to decompose any finite automaton with a strongly connected state diagram into a strongly connected version of what we call a synchronizable cascade decomposition. This is a two component cascade decomposition whose first component has a synchronizer and whose second component is a permutation automaton. Here, we give a simpler procedure for constructing such a decomposition and show that the constructed decomposition is a homomorphic image of a subautomaton of any other synchronizable cascade decomposition. The constructed decomposition is then of minimal size, and all minimal synchronizable cascade decompositions are isomorphic, including all decompositions constructed by the old procedure. This means that their first components are isomorphic, but their second components need not be. In analyzing learning systems, we can use a synchronizable cascade decomposition to model a finite automaton environment, and in these analytical applications, the second component has equiprobable states and can often be ignored in analysis. There are many ways of constructing synchronizable cascade decompositions and we will want to use the construction that is easiest for the analytical application. The isomorphism result says that two different construction methods produce isomorphic decompositions provided the decompositions produced have the minimal number of states, and it is often easy to show this by a simple counting argument. This paper confines itself to giving the simpler procedure and proving the homomorphism result. It does not discuss analytical applications.

Key Words: automaton; decomposition; cascade; synchronizer.

1. INTRODUCTION

In [8] we gave a procedure for decomposing any strongly connected finite automaton into a two component cascade whose first component is an automaton

0022-0000/02 $35.00

with a synchronizer and whose second component is a permutation automaton. The decomposition is useful in studying learning systems that operate in finite automaton environments. In this paper we give a simpler procedure for constructing the decomposition and show the sense in which the constructed decomposition is minimal.

If the components of a finite automaton decomposition have fewer states than the undecomposed automaton, then the decomposition might provide an efficient way of implementing the automaton. Thus much discussion of finite automaton decomposition is motivated by an attempt to achieve a small number of component states [1]. The classical Krohn–Rhodes many-component cascade decomposition is particularly successful in achieving this.

Our decomposition fails where Krohn–Rhodes succeeds. We would not recommend our decomposition as a step toward efficient implementation of an automaton. Our motivation is rather different. We are interested in conceptual decompositions of environments.

We have been studying learning systems that operate in environments that are finite automata [7, 9, 11]. The environment outputs real numbers called payoffs, the current payoff depending only on the current state. The learning system feeds strings of symbols into the finite automaton that is its environment. These strings of input symbols are selected by the learning system probabilistically according to probabilities given by parameters held in the system and varied gradually on the basis of the payoff that is output by the environment. The system varies the parameters according to rules given by the system's reward scheme. Many of the systems we study are "classifier systems," production systems modeled on genetic systems [4–6, 10].

Thus in our view, the automaton is not the system, but is instead the environment of the system. We use the finite automaton formalism to capture certain environment properties (in contrast to the Learning Automata approach in which the formalism is used to capture system properties and it is the system that is called the automaton).

Our decomposition is a conceptual tool for the study of reward schemes. In the remainder of this introduction, we give a brief overview of the use of this tool. The rest of the paper discusses the properties of the decomposition, not its use. A proper discussion of its use can be found in [11].

We can decompose the environment E and then pretend that the first component E_1 of the decomposition is the environment to which the system is trying to adapt and that the second component E_2 is merely a source of noise that modifies the true payoff, defining the true payoff for a state of E_1 as the average payoff for that state, averaged over the states of E_2. Thus, for formal questions in which the noise is not relevant, conclusions for environments with synchronizers are often valid for all environments. In our formulations the states of E_2 are always equiprobable, whatever the action of the system, so E_2 has no memory of past system action. A synchronizer of the first component can therefore be viewed as erasing the environment's memory. Credit for later environment behavior need not be allocated to system actions that were prior to the erasure. Thus it is often easier to answer formal questions about reward allocation if we can rely on the presence of memory

erasing synchronizers. (E.g., see [7], where the use of E_1 in place of E would have simplified the argument.) It is the synchronizers we want, and the number of component states is not terribly important, since we are not implementing the automaton. The automaton is the environment, not the system.

In our work, the environment E is always a strongly connected finite automaton, so it can always be decomposed into two components, E_1 and E_2, in the way discussed here. In [9], we showed how the probability distribution over the input symbols (this distribution being defined by the system) induces probability distributions over the states of E and of E_1, and hence implicitly defines what we call the value of each state of E and E_1. We showed how the value of each input symbol can be naturally defined in terms of the values of the states of E. We showed further that an input symbol has the same value whether that value is defined using E state values or E_1 state values. In its reward scheme, the system needs to increase the probabilities of inputting the more valuable symbols. We have been investigating both global schemes, such as *profit sharing* [2, 7], and local schemes such as *Q-learning* and the *bucket brigade* [6, 12]. The bucket brigade takes advantage of the fact that the input symbol values can be defined in terms of the values of the states of E, or alternatively in terms of the values of the states of certain models of E. The bucket brigade is a mechanism for estimating the values of these states. One usually thinks of a good model of E as a homomorphic image of E, but in [9] we noted that there is a sense in which a homomorphic image of E_1 makes an even better model, since the synchronizers of E_1 ensure that such a model is self correcting.

Output of the environment provides the learning system with two kinds of information: information as to how well the system is doing and information regarding what the current state of the environment is. Output that provides the first we call payoff. This is the raw material of the reward scheme. Output that provides the second I call ordinary output. This is used during performance. For example, in a production system such as a classifier system [6], it is the ordinary output that must satisfy a production's condition.[1] In some situations, an output number can function both as payoff and as ordinary output; the distinction is on the basis of how the system uses the output. A proper discussion of the distinction can be found in [11]. In classifier systems, payoff and ordinary output are traditionally quite different. Strictly speaking, the investigations described in the previous paragraph are of situations where there is no ordinary output, that is, where the system uses environment output only to gauge how well it is doing and not to make inferences regarding current environment state, so what the system knows of environment state is only through knowing which symbols it has input into the environment. This was the starting point of Holland's reward scheme investigations [3].

We have extended our investigations to situations in which the system is a production system that uses ordinary output. The extension for profit sharing is reported in [7], though this does not use our decomposition. We are currently

[1] This assumes no working memory (message list). If there is a working memory then our analysis regards it as part of the environment, so the matched message is part of ordinary output. See [11] for discussion.

examining ways of combining the notion of "prescription" in [7] with the development in [9] to yield a useful analysis of other reward schemes for production systems.

2. SYNCHRONIZABLE CASCADE DECOMPOSITIONS

In this section, we give intuitive definitions of some standard terms. We repeat the definitions more carefully later. (Further discussion of cascade decomposition can be found, for example, in [1].)

In this paper, *cascade* means an automaton with two automaton components, say E_1 and E_2, where in each time unit the input to E_1 is the input to the cascade, and the input to E_2 is a pair consisting of the input to the cascade together with the state of E_1. The automaton E_1 is called the first component and E_2 the second component. We write the cascade automaton as E_1E_2.

A *cascade decomposition* of an automaton E is a cascade E_1E_2 that can be used to simulate E. This means that it has the same input alphabet as E and there is a function h from states of E_1E_2 onto the state set of E such that any input symbol that carries state s to state s' in E_1E_2, also carries state $h(s)$ to state $h(s')$ in E. In that sense, h is a homomorphism.[2] Then if E is in state $\dot{s}$, we can start the simulator E_1E_2 in a state $\ddot{s}$ for which $h(\ddot{s}) = \dot{s}$ and then feed into the simulator the same input symbols that E receives. Then at any future time, we can use h to determine the current state of E from the current state of the simulator. The domain of h must be closed, in the following sense: If E_1E_2 starts in a state that is in the domain, then it will always be in some state in the domain, whatever inputs it receives. The domain might be the whole state set of E_1E_2, but that is not necessary because we can simulate E perfectly well using just the states in the domain and ignoring the others. These states form a *subautomaton* of E_1E_2, so E is a homomorphic image of a subautomaton of E_1E_2.

An automaton is strongly connected if for any ordered pair of states there is an input string that carries the first state to the second. The automaton we are decomposing is supposed to represent an environment in which experiments are repeatable, so we are interested only in decompositions of strongly connected automata.

If there is an input string ρ and a state s such that ρ carries every state to s, then we call ρ a *sychronizer*. A *permutation automaton* is one in which every input symbol induces a permutation on the states. That is, for any input symbol r and any state s, there is a state that r carries to s. In this paper, a *synchronizable cascade decomposition* is a two component cascade decomposition in which the first component has a synchronizer and the second component is a permutation automaton.

[2] Our "homomorphisms" are simply functions from state sets to state sets that "preserve the transition function;" see Definition 4, Section 7.1. Neither are they homomorphisms of the function monoid, nor are they any of the other possible homomorphisms of automaton structure.

3. PURPOSE OF THE PAPER

Different synchronizable cascade decompositions of an automaton E might have different numbers of states. A synchronizable cascade decomposition of E is *minimal* if no synchronizable cascade decomposition of E has fewer states.

We show that if $\bar{E}$ is a minimal synchronizable cascade decomposition of a strongly connected E, and if $\hat{E}$ is any other synchronizable cascade decomposition of E, then $\bar{E}$ is a homomorphic image of a subautomaton of $\hat{E}$. We give a simple concrete procedure that constructs a strongly connected minimal synchronizable cascade decomposition of any strongly connected finite automaton.

It follows that every strongly connected finite automaton has a synchronizable cascade decomposition and that all the minimal ones are isomorphic. Our construction procedure is based on a set of arbitrarily chosen functions. A different choice of functions can often give a different constructed decomposition, but since the different constructed decompositions are all minimal, they are all isomorphic.

Any synchronizable cascade decomposition that has the same number of states as one of our constructed decompositions is isomorphic to it, and hence is strongly connected. It is often easy to count the states of a decomposition. For example, it is easy to see that the synchronizable cascade decomposition constructed in [8] has the same number of states as the decomposition constructed here. The excessively abstract construction procedure given there established the fact that every strongly connected finite automaton has a synchronizable cascade decomposition, but it is now clear that any of the decompositions constructed more simply here is isomorphic to the one constructed there.

Because of the first component synchronizer, two isomorphic synchronizable cascade decompositions must have isomorphic first components, but their second components need not be isomorphic. By using different sets of the arbitrarily chosen functions, our construction procedure here can often construct different second components that are not isomorphic to one another. We will see the sense in which the second components that can be so constructed are the only second components a minimal synchronizable cascade decomposition can have.

I confess that the word "minimal" is motivationally misleading. "Minimal" here means having the fewest states. However, we have noted that there is no particular advantage in having a small number of states, since in the applications we are not building the decomposition and its use is only conceptual. The chief value of the minimality theorem is that it gives us the isomorphism result. In addition, the minimality theorem shows the sense in which the decompositions we construct are the tidiest synchronizable cascade decompositions of strongly connected automata. The isomorphism result allows us to construct these tidy decompositions by whatever method is appropriate to the analysis we are doing, and to know that we end up with essentially the same decomposition, whatever method we use.

4. STRATEGY, TERMINOLOGY, AND NOTATION

We organize our argument as follows. We first give our construction procedure and show that it produces a strongly connected synchronizable cascade decomposi-

tion of any strongly connected finite automaton E. We then prove our minimality theorem: This constructed decomposition is a homomorphic image of a subautomaton of any other synchronizable cascade decomposition of E. Finally, we examine isomorphisms between components of different minimal synchronizable cascade decompositions.

During our argument, we will need a name for the decomposition that our procedure constructs. In this paper, a *constructed decomposition* will mean a decomposition that can be constructed by our procedure. The word *constructed* will have this special meaning. Of course we shall prove that the constructed decompositions are the minimal synchronizable cascade decompositions (up to isomorphism), so we shall then have no more need of the special term "constructed decomposition," but we do need the term during the proof and before.

In this paper, "automaton" means finite automaton. As usual, we require our automata to have one or more states; empty state sets are not allowed. In this paper, a decomposition of an automaton E is required to have the same input alphabet as E. In this paper, cascades and cascade decompositions always have only two components. To say that $\bar{E}$ is a cascade decomposition of E implies of course that there is a homomorphism h from the state set of a subautomaton of $\bar{E}$ onto the state set of E. In this paper, we package that implication by saying that the triple $\langle \bar{E}, h, E \rangle$ is a "cascade simulation." It makes the formal arguments cleaner if we deal always with these triples, with the cascade simulations, and refrain from using the word "decomposition." So in our exposition we shall carefully define from scratch the notion of simulation as a triple, and shall carry through almost the whole exposition using "synchronizable cascade simulations," and using the word "decomposition" only in motivational comments.

The procedure we give will construct a synchronizable cascade simulation and the minimality theorem will say (roughly) that if $\langle E_1 E_2, f, E \rangle$ is a synchronizable cascade simulation and we use our construction procedure to construct a synchronizable cascade simulation $\langle \dot{E}_1 \dot{E}_2, h, E \rangle$, then there is a homomorphism g such that $\langle E_1 E_2, g, \dot{E}_1 \dot{E}_2 \rangle$ is a synchronizable cascade simulation and f is the composition of g and h. At the very end of our discussion, we define "synchronizable cascade decomposition" of E to be a cascade $\bar{E}$ such that there is a synchronizable cascade simulation $\langle \bar{E}, h, E \rangle$. It is then easy to translate the minimality theorem from a statement about cascade simulations to a statement about cascade decompositions.[3]

If f is a function, then we write $Dom(f)$ and $Ran(f)$ to mean its domain and range, respectively. We write function application on the left. If $s \in Dom(f)$ then $f(s)$ is the value of the function when its argument is s. We write function composition from left to right. So if f and g are functions, the composition fg means f followed by g. Thus if $s \in Dom(fg)$, we write $(fg)(s) = g(f(s))$.

If f is a function and $B \subseteq Dom(f)$, then we write $f[B]$ to mean $\{f(s) \mid s \in B\}$, the image of B under f.

[3] In this paper, I use the word "Lemma" to mean fact that is referred to only within the section of the paper in which it is proved, and I use the word "Theorem" to mean fact that is referred to in other sections. I do not mean to imply that the Theorems are more important than the Lemmas.

If ρ is an input string for an automaton, and s is a state of the automaton, then $\rho * s$ is the state to which ρ carries s. This notation is ambiguous whenever s is the state of more than one automaton for which ρ is an input string. In that case, we shall be careful to say which automaton we are talking about, unless the state to which ρ carries s in every one of these is the same state. When it is clear what automaton we are talking about, we refer to $*$ as the transition function of that automaton.

Of course any input string induces a function on the set of states. We will often use the name of the string as the name of the function. That is, if ρ is an input string and S is the set of states, we can say $\rho: S \to S$, and if $s \in S$ we can write $\rho(s) = \rho * s$, where on the left, the "ρ" is the function induced by the string ρ. Similarly, if $B \subseteq S$ we can write, $\rho[B] = \{\rho * s \mid s \in B\}$. Also, $Ran(\rho) = \{\rho * s \mid s \in S\}$. Again, we shall be careful to say which automaton we are talking about if otherwise there would be ambiguity.

5. PLAN OF THE PAPER

Our construction is built fundamentally on minimal size ranges of functions induced by input strings. Section 6 investigates minimal size ranges and proves two fundamental theorems used in practically all subsequent sections. Section 7 formally introduces simulations and investigates the images of minimal size ranges under a simulation's homomorphism. It then discusses strong connectivity in simulations, and what the presence of a synchronizer implies in simulations. Section 8 introduces synchronizable cascades. It examines the form of their minimal size ranges. It then defines the important notion of "synchronizable cascade simulation."

We are then ready to look at our construction procedure. Section 9 gives the procedure and proves that it constructs a synchronizable cascade simulation. Section 10 then proves the minimality theorem, except that it is stated there in terms of simulations rather than decompositions. Section 11 finally defines decomposition and translates the minimality theorem into a statement about decompositions.

A general discussion of the implications of the minimality theorem begins in the last part of Section 10 and continues into Section 11.

6. MINIMAL SIZE RANGES

Let E be a finite automaton with state set S. In Section 9 we will describe our procedure for constructing a synchronizable cascade decomposition of E. The states of the first component of the decomposition will be what we call minimal size ranges of E.

DEFINITION 1 (Minimal Size Range). Look at each input string ρ and regard each as a function on the state set S. For each ρ, look at the size of its range. Let n be the smallest integer for which there is a range of size n. A **minimal size range** is a size n range.

This is the logical place to give a definition that we shall not use until Section 8.

DEFINITION 2 (Permutation Automaton). An automaton is a **permutation automaton** provided every input symbol (and consequently every input string) induces a one to one function.

So if E were a permutation automaton, there would be only one minimal size range, namely the entire state set S. In fact, S would be the only range.

We now prove two theorems we will need in constructing the transition function of the first component of our decomposition.

THEOREM 6.1. *If ρ is an input string and $Ran(\rho)$ is a minimal size range, then the function ρ maps every minimal size range one to one and onto $Ran(\rho)$.*

Proof. Let $Ran(\tau)$ be a minimal size range, where τ is some input string. Then $Ran(\tau\rho) \subseteq Ran(\rho)$. Since $Ran(\rho)$ is minimal size, $Ran(\tau\rho) = Ran(\rho)$. But $\rho[Ran(\tau)] = Ran(\tau\rho)$, so $\rho[Ran(\tau)] = Ran(\rho)$. Since $Ran(\tau)$ and $Ran(\rho)$ are the same size, ρ maps $Ran(\tau)$ one to one and onto $Ran(\rho)$. ▮

THEOREM 6.2. *Consider any minimal size range. The set of its images under functions induced by the various input strings is exactly the set of all minimal size ranges.*

Proof. Suppose $Ran(\rho)$ is a minimal size range, for some input string ρ. We first show that every image of $Ran(\rho)$ is a minimal size range. An image of $Ran(\rho)$ is also a range since $\tau[Ran(\rho)] = Ran(\rho\tau)$. Obviously it is no larger than $Ran(\rho)$.

We now show that every minimal size range is an image of $Ran(\rho)$. Suppose $Ran(\tau)$ is a minimal size range. Then by Theorem 6.1, the function τ maps $Ran(\rho)$ onto this range. ▮

7. SIMULATIONS

7.1. Homomorphic Images of Minimal Size Ranges

Our minimality theorem says that our constructed decomposition is a certain homomorphic image. To prove it is, we need to show that homomorphisms map minimal size ranges to minimal size ranges. We do that in this subsection.

If we have two states of a finite automaton, we say that the second state is **accessible** from the first provided there is an input string that carries the automaton from the first state to the second. Note that "is accessible from" is a transitive relation.

DEFINITION 3 (Closed). If E is a finite automaton with state set S and if $Q \subseteq S$, then we say Q is **closed** if no states outside Q are accessible from states inside Q.[4]

DEFINITION 4 (Homomorphism). Suppose $\bar{E}$ and E are two automata with the same input alphabet and with state sets $\bar{S}$ and S, respectively. We call h a **homomorphism linking $\bar{E}$ to E** if h is a function from a closed subset of $\bar{S}$ into S, and if for any input symbol r and for any $\bar{s} \in Dom(h)$, the equality $h(r * \bar{s}) = r * h(\bar{s})$ holds.

Clearly the $*$ on the left is the transition function of $\bar{E}$, and the $*$ on the right is the transition function of E. Note that $Dom(h)$ is a closed subset of $\bar{S}$, and h: $Dom(h) \to S$. Note also that, since h need not be onto, $Dom(h)$ could be empty.

[4] So $\varnothing$ is closed.

It follows by induction on string length that if h is such a homomorphism, then for any $\bar{s} \in Dom(h)$ and any input string ρ, we have $\rho * h(\bar{s}) = h(\rho * \bar{s})$. In other words, if $\bar{s} \in Dom(h)$ then

$$\rho(h(\bar{s})) = h(\rho(\bar{s})). \tag{1}$$

Then clearly, if $\bar{R} \subseteq Dom(h)$, we have

$$\rho[h[\bar{R}]] = h[\rho[\bar{R}]]. \tag{2}$$

Of course, in Eqs. (1) and (2), the ρ on the left is a function on the state set of E, and the ρ on the right is a function on the state set of $\bar{E}$.

We will use the following obvious theorem in the next subsection.

THEOREM 7.1. *The range of any homomorphism is closed.*[5]

Proof. Suppose $\bar{E}$ and E are automata with the same input alphabet, and suppose h is a homomorphism linking $\bar{E}$ to E. Suppose $s \in Ran(h)$ and suppose $\dot{s}$ is a state of E accessible from s. We need to show $\dot{s} \in Ran(h)$. Let ρ be an input string that carries s to $\dot{s}$ in E. Let $\bar{s}$ be a state of $\bar{E}$ such that $h(\bar{s}) = s$. Then the left side of Eq. (1) is $\dot{s}$, so $\dot{s} \in Ran(h)$. ▮

DEFINITION 5 (Simulation). In this paper, a **simulation** is a triple $\langle \bar{E}, h, E \rangle$, where $\bar{E}$ and E are finite automata with the same input alphabet, h is a homomorphism linking $\bar{E}$ to E, and the range of the function h is the whole state set of E. (Intuitively, $\bar{E}$ is simulating E.) We sometimes call it a simulation "of E."[6]

DEFINITION 6 (Subautomaton). Suppose S and $\bar{S}$ are state sets of automata E and $\bar{E}$, and suppose f is the identity function on S. Then E is a **subautomaton** of $\bar{E}$, provided $S \subseteq \bar{S}$, and $\langle \bar{E}, f, E \rangle$ is a simulation.

If E is a subautomaton of $\bar{E}$, then of course the state set of E is a closed subset of the state set of $\bar{E}$. Note that in an automaton $\bar{E}$, any non-empty closed subset of states forms the state set of a subautomaton.

In the remainder of this subsection, we assume we have a simulation $\langle \bar{E}, h, E \rangle$ for which $Dom(h)$ is the whole state set of $\bar{E}$. The state sets of E and $\bar{E}$ will be S and $\bar{S}$, respectively. We prove an uninspiring but key theorem, which says how minimal size ranges behave in such a simulation.

LEMMA 7.2. *If ρ is an input string, then $Ran(\rho)$ in E is the image under h of $Ran(\rho)$ in $\bar{E}$.*

Proof. $h[\bar{S}] = S$, so the lemma follows from Eq. (2) by setting $\bar{R} = \bar{S}$ and noting that $\rho[\bar{S}]$ is $Ran(\rho)$ in $\bar{E}$ and that $\rho[S]$ is $Ran(\rho)$ in E. ▮

LEMMA 7.3. *The image under h of a minimal size range is itself a minimal size range.*

[5] Of course the range can be empty.

[6] So $Dom(h) \neq \varnothing$.

Proof. Let $Ran(\rho)$ be a minimal size range of $\bar{E}$. We need to show that the image $h[Ran(\rho)]$ is a minimal size range of E. By Lemma 7.2, the image is $Ran(\rho)$ in E. We show it is minimal size. Let τ be any input string. We show that in E, the range $Ran(\rho)$ is no larger than $Ran(\tau)$. In $\bar{E}$, we have $Ran(\tau\rho) \subseteq Ran(\rho)$, and since $Ran(\rho)$ is minimal size, $Ran(\tau\rho) = Ran(\rho)$. Applying h to both sides, and using Lemma 7.2, tells us that in E, $Ran(\tau\rho) = Ran(\rho)$, or $\rho[Ran(\tau)] = Ran(\rho)$. So in E, $Ran(\rho)$ is the image of $Ran(\tau)$ under some function (in this case, under ρ) and so $Ran(\rho)$ is no larger than $Ran(\tau)$. ▮

LEMMA 7.4. *Any minimal size range of E is the image under h of some minimal size range of $\bar{E}$.*

Proof. Let Q be a minimal size range of E. Take any minimal size range $\bar{R}$ of $\bar{E}$. Then by Lemma 7.3, $h[\bar{R}]$ is a minimal size range of E. By Theorem 6.2, there is an input string ρ such that $\rho[h[\bar{R}]] = Q$. By Theorem 6.2, if we look at images of ranges under functions induced by input strings, images of minimal size ranges are minimal size ranges. Thus in $\bar{E}$, the set $\rho[\bar{R}]$ is a minimal size range. But by Eq. (2), we have $h[\rho[\bar{R}]] = \rho[h[\bar{R}]] = Q$, so Q is the image under h of the minimal size range $\rho[\bar{R}]$. ▮

We combine these last two lemmas into

THEOREM 7.5. *If $\langle \bar{E}, h, E \rangle$ is a simulation and the domain of h is the whole state set of $\bar{E}$, then the images under h of the minimal size ranges of $\bar{E}$ are exactly the minimal size ranges of E.*

Proof. This is from Lemma 7.3 and Lemma 7.4. ▮

7.2. Strong Connectivity

So far we have not used the fact that our environment E is strongly connected. Since it is, we can clean up our simulations as we describe in this subsection.

DEFINITION 7 (Strongly Connected). Suppose E is an automaton with state set S and suppose $Q \subseteq S$. Then we say Q is **strongly connected** if every state of Q is accessible from every other state of Q. If S is strongly connected, then we say the automaton E is strongly connected.[7]

Note that if E is a strongly connected automaton then its only subautomaton is itself and the only non-empty closed set of states is the set of all states.

LEMMA 7.6. *If E is a finite automaton then there is a strongly connected subautomaton of E.*

Proof. For each state s of E we define the accessibility number of s to be the number of states accessible from s (including s itself). We find the lower bound N of the accessibility numbers. Let s be a state with accessibility number N and let $\bar{S}$ be the set of N states accessible from s. Consider an arbitrary $\bar{s} \in \bar{S}$. A state outside $\bar{S}$ can't be accessible from $\bar{s}$, or it would be accessible from s. Every state inside $\bar{S}$

[7] So $\varnothing$ is strongly connected.

must be accessible from $\bar{s}$ since its accessibility number can't be less than N. Thus $\bar{S}$ is closed and strongly connected. Thus it forms the state set of a strongly connected subautomaton of E. ▌

DEFINITION 8 (Clean Simulation). If $\langle \bar{E}, h, E \rangle$ is a simulation then the domain of h is closed, by the definition of homomorphism. If, in addition, the domain of h is strongly connected, then we call the simulation a **clean simulation**.

If the simulation is clean, then E must be strongly connected. Of course, if $\bar{E}$ is strongly connected, then the simulation must be clean, and the domain of h is the whole state set of $\bar{E}$. More generally, we have the following theorem.

THEOREM 7.7. *If $\langle \bar{E}, h, E \rangle$ is a simulation and E is strongly connected, then there is a clean simulation $\langle \bar{E}, f, E \rangle$.*

Proof. Let S be the state set of E. By the definition of simulation, $Dom(h)$ is closed. Let $\dot{E}$ be the automaton formed from $\bar{E}$ by throwing away states outside $Dom(h)$. By Lemma 7.6, there is a strongly connected subautomaton $\ddot{E}$ of $\dot{E}$. Let $\ddot{Q}$ be the state set of $\ddot{E}$ and let f be the restriction of h to $\ddot{Q}$. Then $f: \ddot{Q} \to S$ is a homomorphism, and $\ddot{Q}$ is non-empty, closed, and strongly connected. By Theorem 7.1, $Ran(f)$ is closed. Since S is strongly connected, $Ran(f) = S$. ▌

We shall use Theorem 7.7 in translating the minimality theorem from a statement about simulations to a statement about decompositions.

7.3. Synchronizers

The first component of our constructed decomposition will have a synchronizer. This constrains homomorphisms of the first component. In this subsection we examine this constraint.

DEFINITION 9 (Synchronizer). We call an input string σ a **synchronizer** of automaton E (or merely a synchronizer) if there is a state s of automaton E such that whatever the current state of E, the state of E after the input of σ is guaranteed to be s. (We say σ synchronizes E to s.)

Note that an automaton E has a synchronizer if and only if there is an input string whose range is a singleton. If E is strongly connected and has a synchronizer, then for each state of E there is a synchronizer that synchronizes E to that state.

The following theorem will be useful when we have proved the minimality theorem and are discussing its implications.

THEOREM 7.8. *If $\langle \bar{E}, h, E \rangle$ and $\langle \bar{E}, g, E \rangle$ are clean simulations, and if $\bar{E}$ has a synchronizer, then $h = g$.*

Proof. Let $\bar{S}$ and S be the state sets of $\bar{E}$ and E, respectively. Let us define a subset $\bar{Q}$ of $\bar{S}$ as follows. $\bar{s} \in \bar{Q}$ if and only if there is a synchronizer that synchronizes $\bar{E}$ to $\bar{s}$. Then clearly $\bar{Q}$ is non-empty, closed, and strongly connected. $\bar{Q}$ is a subset of every non-empty closed subset of $\bar{S}$. Also, if $\bar{Q} \subseteq \bar{B}$ and $\bar{B} \subseteq \bar{S}$, and if $\bar{B}$ is strongly connected, then $\bar{B} = \bar{Q}$. Thus we see that $\bar{Q}$ is the only non-empty closed

strongly connected subset of $\bar{S}$. Since $Dom(h)$ and $Dom(g)$ are both non-empty, closed, and strongly connected, $Dom(h) = Dom(g) = \bar{Q}$.

Select any $\bar{s} \in \bar{Q}$. There is some input string ρ that synchronizes $\bar{E}$ to $\bar{s}$. Then since h is a homomorphism, ρ synchronizes E to $h(\bar{s})$, and since g is a homomorphism, ρ synchronizes E to $g(\bar{s})$. So $h(\bar{s}) = g(\bar{s})$. ▮

8. SYNCHRONIZABLE CASCADES

8.1. Minimal Size Ranges of Synchronizable Cascades

In this section we define "synchronizable cascade" and prove a key theorem regarding the form of the minimal size ranges of its subautomatons.

DEFINITION 10 (Cascade). In this paper, **cascade** means an automaton with two automaton components, say E_1 and E_2, where in each time unit, the input to E_1 is the input to the cascade, and the input to E_2 is a pair consisting of the input to the cascade together with the state of E_1. The automaton E_1 is called the first component and E_2 the second component.

We write the cascade automaton as E_1E_2. If the state set of E_1 is S_1 and of E_2 is S_2, then the state set of E_1E_2 is $S_1 \times S_2$.

Suppose r is an input symbol which we can input to E_1E_2. First let us look at E_1. If $s_1 \in S_1$, we write $r * s_1$ to mean the member of S_1 to which r carries s_1. Now look at E_2. The input to E_2 is a pair $\langle r, s_1 \rangle$, where $s_1 \in S_1$. If $s_2 \in S_2$, we write $\langle r, s_1 \rangle * s_2$ to mean the member of S_2 to which $\langle r, s_1 \rangle$ carries s_2. Now look at E_1E_2. If $\langle s_1, s_2 \rangle \in S_1 \times S_2$, we write $r * \langle s_1, s_2 \rangle$ to mean the member of $S_1 \times S_2$ to which r carries $\langle s_1, s_2 \rangle$. Now E_1E_2 is a cascade, so for $s_1 \in S_1$, $s_2 \in S_2$, and r an input symbol, we have

$$r * \langle s_1, s_2 \rangle = \langle r * s_1, \langle r, s_1 \rangle * s_2 \rangle. \tag{3}$$

This equation characterizes the notion of cascade.

DEFINITION 11 (Synchronizable Cascade). A **synchronizable cascade** is a cascade whose first component has a synchronizer and whose second component is a permutation automaton.

DEFINITION 12 (Strongly Connected Synchronizable Cascade). A **strongly connected synchronizable cascade** is a synchronizable cascade that is strongly connected.

In the rest of this section, E_1E_2 will be a synchronizable cascade.

If ρ is an input string of E_1E_2, then we can regard ρ as a function $\rho: S_1 \times S_2 \to S_1 \times S_2$, and of course we can write $\rho(s_1, s_2) = \rho * \langle s_1, s_2 \rangle$. Whenever $B \subseteq S_1 \times S_2$, we can examine the image $\rho[B]$. We now examine certain such images.

LEMMA 8.1. *If r is an input symbol and $s_1 \in S_1$, then $r[\{s_1\} \times S_2] = \{r * s_1\} \times S_2$.*

Proof. Suppose $r(s_1, s_2) = r(s_1, s_2')$ in E_1E_2. That is, suppose $r * \langle s_1, s_2 \rangle = r * \langle s_1, s_2' \rangle$. Then by Eq. (3), $\langle r, s_1 \rangle * s_2 = \langle r, s_1 \rangle * s_2'$. Then since E_2 is a permuta-

tion automaton, we have $s_2 = s_2'$, and consequently $\langle s_1, s_2\rangle = \langle s_1, s_2'\rangle$. So we have shown $r(s_1, s_2) = r(s_1, s_2') \Rightarrow \langle s_1, s_2\rangle = \langle s_1, s_2'\rangle$.

In other words, we have shown that the function r restricted to $\{s_1\} \times S_2$ is a one to one function. It follows that the image $r[\{s_1\} \times S_2]$ is the same size as $\{s_1\} \times S_2$, which is the same size as $\{r * s_1\} \times S_2$. Since clearly $r[\{s_1\} \times S_2] \subseteq \{r * s_1\} \times S_2$, the lemma follows from the size considerations. ▮

LEMMA 8.2. *If ρ is an input string of E_1E_2 and $s_1 \in S_1$, then $\rho[\{s_1\} \times S_2] = \{\rho * s_1\} \times S_2$, where the $*$ is the transition function of E_1.*

Proof. By induction on the length of ρ, use Lemma 8.1 for the length 1 case. ▮

LEMMA 8.3. *If ρ is an input string of E_1E_2 and $s_1 \in S_1$, then the function ρ restricted to $\{s_1\} \times S_2$ is one to one.*

Proof. Let ρ be an input string. By Lemma 8.2, $\rho[\{s_1\} \times S_2] = \{\rho * s_1\} \times S_2$. Hence the restriction has a range that is the same size as its domain and so the restriction must be one to one. ▮

Now let $\bar{E}$ be a subautomaton of the synchronizable cascade E_1E_2 and let Q be its state set. Thus Q is a closed subset of $S_1 \times S_2$. Our key lemma here characterizes the minimal size ranges of $\bar{E}$.

LEMMA 8.4. *Every minimal size range of $\bar{E}$ is of form $(\{s_1\} \times S_2) \cap Q$, for some $s_1 \in S_1$. If, in addition, $\bar{E}$ is strongly connected, then every non-empty state set of form $(\{s_1\} \times S_2) \cap Q$ is a minimal size range of $\bar{E}$.*

Proof. Let us call a subset of $S_1 \times S_2$ *friendly* if it is of form $(\{s_1\} \times S_2) \cap Q$ for some $s_1 \in S_1$.

First we examine E_1E_2. We look at an arbitrary input string ρ of E_1E_2 and note that $\rho[Q] \subseteq Q$. We now show that $\rho[Q]$ is no smaller than the largest friendly sets. Lemma 8.3 says that ρ restricted to $\{s_1\} \times S_2$ is one to one, so ρ restricted to the friendly set $(\{s_1\} \times S_2) \cap Q$ is also one to one and we see that $\rho[(\{s_1\} \times S_2) \cap Q]$ is the same size as $(\{s_1\} \times S_2) \cap Q$. But clearly $\rho[(\{s_1\} \times S_2) \cap Q] \subseteq \rho[Q]$, so $\rho[Q]$ is no smaller than $(\{s_1\} \times S_2) \cap Q$. The state s_1 was arbitrary, so we see that in E_1E_2, the image $\rho[Q]$ is no smaller than the largest friendly sets.

We now examine $\bar{E}$. The non-empty friendly sets form a partition of Q, the state set of $\bar{E}$. We see that the function ρ in $\bar{E}$ is simply the restriction to Q of the function ρ in E_1E_2. Thus the conclusion of the last paragraph holds in $\bar{E}$ too: In $\bar{E}$, the image $\rho[Q]$ is no smaller than the largest friendly sets. But in $\bar{E}$, the image $\rho[Q]$ is $Ran(\rho)$. Therefore in $\bar{E}$, the range of any input string ρ cannot be smaller than the largest friendly sets.

Now consider an input string σ that synchronizes E_1 to state $\dot{s}_1$. Then in $\bar{E}$, the range $Ran(\sigma)$ is a subset of the friendly set $(\{\dot{s}_1\} \times S_2) \cap Q$. Since it cannot be smaller than even the largest friendly sets, we see that $Ran(\sigma)$ is a friendly set of the largest size, and hence is a minimal size range of $\bar{E}$.

We now look in $\bar{E}$ at images of sets under functions induced by input strings. Clearly an image of a friendly set is a subset of some friendly set. Now let B be any minimal size range of $\bar{E}$. By Theorem 6.2, B is an image of the friendly set $Ran(\sigma)$,

so B is a subset of some friendly set. But we said that no range can be smaller than the largest friendly sets, and B is a range. So B is the whole of a friendly set. Thus we see that every minimal size range is a friendly set.

Suppose now that $\bar{E}$ is strongly connected. Then the images of $Ran(\sigma)$ cover Q. By Theorem 6.2, these images are minimal size ranges of $\bar{E}$ and hence they are friendly sets. Since they cover Q, they are all the non-empty friendly sets. So all non-empty friendly sets are minimal size ranges of $\bar{E}$. ▮

THEOREM 8.5. *Suppose $\bar{E}$ is a strongly connected subautomaton of a synchronizable cascade E_1E_2, and suppose Q and $S_1 \times S_2$ are the state sets of $\bar{E}$ and E_1E_2, respectively. Then the minimal size ranges of $\bar{E}$ are exactly the non-empty sets of form $(\{s_1\} \times S_2) \cap Q$, for $s_1 \in S_1$.*

Proof. Direct from Lemma 8.4. ▮

COROLLARY 8.6. *If E_1E_2 is a strongly connected synchronizable cascade, then the minimal size ranges are exactly the sets of form $\{s_1\} \times S_2$.*

Proof. This is from Theorem 8.5 with $\bar{E} = E_1E_2$. ▮

8.2. Synchronizable Cascade Simulations

DEFINITION 13 (Various Cascade Simulations). A simulation $\langle \bar{E}, h, E \rangle$ is called a **cascade simulation** (of E) if $\bar{E}$ is a cascade, a **synchronizable cascade simulation** if $\bar{E}$ is a synchronizable cascade, and a **clean synchronizable cascade simulation** if $\bar{E}$ is a synchronizable cascade and the simulation is clean.

The purpose of this paper is to examine synchronizable cascade simulations $\langle E_1E_2, g, E \rangle$ in which E is strongly connected. We begin by examining in this section the simple case in which E is itself a synchronizable cascade $\dot{E}_1\dot{E}_2$. Then, by Theorem 7.7, there is a homomorphism h such that $\langle E_1E_2, h, \dot{E}_1\dot{E}_2 \rangle$ is a clean synchronizable cascade simulation. We said in the Introduction that in applications we often regard the second components as merely noise and concentrate on the first components. We now prove a theorem that indicates how this can be tidily done in the above simulation. The theorem will be useful when we have proved the minimality theorem and are discussing its implications. We begin with a definition.

DEFINITION 14 (Collapse). Suppose we have a function $f: A \to B$ and suppose that A and B are each sets of ordered pairs. Suppose A_1 is the set of all first entries of the pairs in A, and B_1 is the set of all first entries of the pairs in B. We say f **collapses onto** f_1 if the function $f_1: A_1 \to B_1$ is such that $f(x, y) = \langle w, z \rangle \Rightarrow f_1(x) = w$, for any $\langle x, y \rangle \in A$ and $\langle w, z \rangle \in B$.

THEOREM 8.7. *Suppose $\langle E_1E_2, h, \dot{E}_1\dot{E}_2 \rangle$ is a clean synchronizable cascade simulation, and $\dot{E}_1\dot{E}_2$ is a strongly connected synchronizable cascade. Then h collapses onto some function h_1, and $\langle E_1, h_1, \dot{E}_1 \rangle$ is a clean simulation.*

Proof. Suppose $\langle E_1E_2, h, \dot{E}_1\dot{E}_2 \rangle$ is a clean synchronizable cascade simulation, and $\dot{E}_1\dot{E}_2$ is a strongly connected synchronizable cascade. Let $S_1 \times S_2$ and $\dot{S}_1 \times \dot{S}_2$ be the state sets of E_1E_2 and $\dot{E}_1\dot{E}_2$, respectively. Let Q be the domain of h. So Q is

closed and strongly connected. Let $\bar{E}$ be the strongly connected automaton formed from E_1E_2 by removing all states outside Q. Let Q_1 be the set of all first entries in pairs in Q. So $Q_1 \subseteq S_1$.

We first show Q_1 is closed. Suppose $s_1 \in Q_1$ and suppose r is any input symbol. Then there is a state s_2 of E_2 such that $\langle s_1, s_2 \rangle \in Q$. Then since Q is closed, $r * \langle s_1, s_2 \rangle \in Q$, and by Eq. (3), $r * s_1 \in Q_1$. So Q_1 is closed.

$\langle \bar{E}, h, \dot{E}_1\dot{E}_2 \rangle$ is a simulation, and so, by Theorem 8.5, Theorem 7.5, and Corollary 8.6, we see that for any $s_1 \in Q_1$, the set $h[(\{s_1\} \times S_2) \cap Q]$ is of form $\{\dot{s}_1\} \times \dot{S}_2$ for some $\dot{s}_1 \in \dot{S}_1$. We define $h_1 : Q_1 \to \dot{S}_1$ so that for any $s_1 \in Q_1$, the value $h_1(s_1)$ is that very $\dot{s}_1$. In other words, $h[(\{s_1\} \times S_2) \cap Q] = \{h_1(s_1)\} \times \dot{S}_2$. Thus $h(s_1, s_2) = \langle \dot{s}_1, \dot{s}_2 \rangle \Rightarrow h_1(s_1) = \dot{s}_1$, and we see that h collapses onto h_1.

We now need to show that h_1 is a homomorphism. Select an arbitrary $s_1 \in Q_1$ and an arbitrary input symbol r. We need to show $h_1(r * s_1) = r * h_1(s_1)$. Select some s_2 such that $\langle s_1, s_2 \rangle \in Q$. Let $\langle \dot{s}_1, \dot{s}_2 \rangle = h(s_1, s_2)$. Since h collapses onto h_1, we have $h_1(s_1) = \dot{s}_1$. Since h is a homomorphism, we have $h(r * \langle s_1, s_2 \rangle) = r * \langle \dot{s}_1, \dot{s}_2 \rangle$, and using Eq. (3) on this gives $h(r * s_1, \langle r, s_1 \rangle * s_2) = \langle r * \dot{s}_1, \langle r, \dot{s}_1 \rangle * \dot{s}_2 \rangle$. So since h collapses onto h_1, we have $h_1(r * s_1) = r * \dot{s}_1 = r * h_1(s_1)$. So h_1 is a homomorphism.

h is onto $\dot{S}_1 \times \dot{S}_2$ by the definition of simulation. Hence h_1 is onto $\dot{S}_1$.

So $\langle E_1, h_1, \dot{E}_1 \rangle$ is a simulation. To show it is clean, we merely need to show that Q_1 is strongly connected.

Q_1 is strongly connected since if s_1 and $\hat{s}_1$ are members of Q_1, there are members s_2 and $\hat{s}_2$ of E_2 such that $\langle s_1, s_2 \rangle \in Q$ and $\langle \hat{s}_1, \hat{s}_2 \rangle \in Q$. Since Q is strongly connected, there is an input string that carries $\langle s_1, s_2 \rangle$ to $\langle \hat{s}_1, \hat{s}_2 \rangle$ in the cascade E_1E_2, and therefore that string carries s_1 to $\hat{s}_1$ in E_1. ▌

Theorem 7.8 tells us that the h_1 in the theorem is the unique function that makes $\langle E_1, h_1, \dot{E}_1 \rangle$ a clean simulation.

9. OUR CONSTRUCTION PROCEDURE

We now give our procedure that takes any strongly connected automaton E and constructs a synchronizable cascade simulation $\langle E_1E_2, h, E \rangle$. Let S be the state set of E. The domain of h will be the whole of $S_1 \times S_2$, the state set of E_1E_2.

The first component E_1 is very simple. Its state set S_1 is the set of all minimal size ranges of E. Theorem 6.2 says that, in E, any input symbol r maps any minimal size range onto another minimal size range. That is, if s_1 is a minimal size range of E, then so is the image $r[s_1]$, where this r is the function induced by the symbol r on the states of E. So the transition function of E_1 is the obvious:

$$r * s_1 = r[s_1]. \tag{4}$$

I will describe the construction of E_2 and h twice, first intuitively and second formally. I hope the intuitive description makes this simple construction obvious, but in any case, the formal description is complete and independent of the intuitive description.

S_2 can be any set that is the same size as a minimal size range of E. Let n be the number of members in a minimal size range. We will construct the function

h: $S_1 \times S_2 \to S$ to be such that $h(s_1, s_2)$ is a state in the minimal size range s_1. Which one? There are n possibilities, and s_2 tells us which one we have. There are n possible values of s_2.

So for each minimal size range s_1 we have a one to one correspondence between s_1 and S_2. If s corresponds with s_2 (where $s \in s_1$ and $s_2 \in S_2$) then $s = h(s_1, s_2)$. We have a separate correspondence for each s_1.

It is obvious that these correspondences can be arbitrary. Given a set of correspondences (one for each minimal size range), we can construct h based on them and then construct the transition function of E_2 to make h a homomorphism in the obvious way.

The formal description of the construction is as follows.

The set S_2 is any set the same size as a minimal size range of E. We begin with an arbitrary set of correspondences, one for each s_1 in S_1,

$$\varphi_{s_1}: s_1 \xrightarrow[onto]{1-1} S_2. \tag{5}$$

We define h: $S_1 \times S_2 \to S$ by

$$h(s_1, s_2) = \varphi_{s_1}^{-1}(s_2). \tag{6}$$

The transition function of E_2 is given by

$$\langle r, s_1 \rangle * s_2 = \varphi_{r * s_1}(r * h(s_1, s_2)), \tag{7}$$

and the transition function of $E_1 E_2$ is given by

$$r * \langle s_1, s_2 \rangle = \langle r * s_1, \langle r, s_1 \rangle * s_2 \rangle. \tag{8}$$

We now give a formal proof that our constructed $\langle E_1 E_2, h, E \rangle$ is a synchronizable cascade simulation.

LEMMA 9.1. *h is a homomorphism.*

Proof. From Eqs. (8) and (7), we have $r * \langle s_1, s_2 \rangle = \langle r * s_1, \varphi_{r * s_1}(r * h(s_1, s_2)) \rangle$. From this and Eq. (6), we have $h(r * \langle s_1, s_2 \rangle) = \varphi_{r * s_1}^{-1}(\varphi_{r * s_1}(r * h(s_1, s_2))) = r * h(s_1, s_2)$. ∎

LEMMA 9.2. *$\langle E_1 E_2, h, E \rangle$ is a cascade simulation.*

Proof. We have seen that $E_1 E_2$ is a cascade. Lemma 9.1 tells us that h is a homomorphism. It only remains to show that h is onto S. Select an arbitrary member s of S. Since as we said, the minimal size ranges (the members of S_1) cover S, we can find an $s_1 \in S_1$ for which $s \in s_1$, and then, by Eq. (6), $h(s_1, \varphi_{s_1}(s)) = s$. So $s \in Ran(h)$. ∎

We note that by induction we can extend Eq. (4). If ρ is an input string of E, and hence an input string of E_1, then for any $s_1 \in S_1$, we have

$$\rho * s_1 = \rho[s_1], \tag{9}$$

where $*$ is the transition function in E_1, and the ρ on the right is the function induced by the string in E.

We now need to show that E_1E_2 is a synchronizable cascade.

LEMMA 9.3. *If $s_1 \in S_1$ then there is an input string ρ that synchronizes E_1 to s_1.*

Proof. By the definition of S_1, the set s_1 is a minimal size range in E. Let ρ be a string such that $Ran(\rho) = s_1$ in E. Let s_1' be an arbitrary state of S_1. Equation (9) tells us that ρ carries this state to $\rho[s_1']$. But by Theorem 6.1, $\rho[s_1'] = s_1$. ▮

LEMMA 9.4. *E_2 is a permutation automaton.*

Proof. From (5) and Eq. (4), we see that if $s_1 \in S_1$, then $\varphi_{r*s_1}[r[s_1]] = S_2$. Also, $\varphi_{s_1}^{-1}[S_2] = s_1$. Thus we can write $\varphi_{r*s_1}[r[\varphi_{s_1}^{-1}[S_2]]] = S_2$. From Eqs. (7) and (6), we have $\langle r, s_1\rangle * s_2 = \varphi_{r*s_1}(r * \varphi_{s_1}^{-1}(s_2))$. Thus in E_2, the input $\langle r, s_1\rangle$ maps the set S_2 onto the image set $\varphi_{r*s_1}[r[\varphi_{s_1}^{-1}[S_2]]]$. We have seen that this image set is S_2. ▮

So E_1E_2 is a synchronizable cascade.

LEMMA 9.5. *E_1E_2 is strongly connected.*

Proof. Suppose $\langle \hat{s}_1, \hat{s}_2\rangle$ and $\langle s_1, s_2\rangle$ are states of E_1E_2. We seek an input string that carries $\langle \hat{s}_1, \hat{s}_2\rangle$ to $\langle s_1, s_2\rangle$ in E_1E_2.

Since the states of E_1 are the minimal size ranges of E, the set s_1 is a minimal size range. Let ρ be an input string such that $Ran(\rho) = s_1$. By (5), there is a state s in s_1 such that $\varphi_{s_1}(s) = s_2$. Then by Eq. (6), we have $h(s_1, s_2) = s$. Furthermore, $s \in s_1$, or in other words, $h(s_1, s_2) \in Ran(\rho)$. So there is a state $\tilde{s}$ in S such that ρ carries $\tilde{s}$ to $h(s_1, s_2)$.

Since E is strongly connected, there is an input string that carries $h(\hat{s}_1, \hat{s}_2)$ to $\tilde{s}$ in E. Let τ be such a string. We shall show that $\tau\rho$ is the string we seek.

We know that $Ran(\tau\rho) \subseteq Ran(\rho)$ in E, and $Ran(\rho)$ is minimal size, so $Ran(\tau\rho) = Ran(\rho) = s_1$. Thus we see by Theorem 6.1 that, in E, the string $\tau\rho$ maps every minimal size range onto s_1. So in E_1 (by Eq. (9)), the string $\tau\rho$ carries every state to s_1. Thus we see that, in E_1E_2, the string $\tau\rho$ carries $\langle \hat{s}_1, \hat{s}_2\rangle$ to $\langle s_1, \ddot{s}_2\rangle$, for some $\ddot{s}_2 \in S_2$. We need only show that $\ddot{s}_2 = s_2$.

Since h is a homomorphism, $\tau\rho$ carries $h(\hat{s}_1, \hat{s}_2)$ to $h(s_1, \ddot{s}_2)$ in E. But we have seen that, in E, the string τ carries $h(\hat{s}_1, \hat{s}_2)$ to $\tilde{s}$, and ρ carries $\tilde{s}$ to $h(s_1, s_2)$. Thus $h(s_1, \ddot{s}_2) = h(s_1, s_2)$. So by Eq. (6), we have $\varphi_{s_1}^{-1}(\ddot{s}_2) = \varphi_{s_1}^{-1}(s_2)$, and since $\varphi_{s_1}^{-1}$ is one to one, $\ddot{s}_2 = s_2$. ▮

THEOREM 9.6. *Our construction procedure takes any strongly connected automaton E and constructs a synchronizable cascade simulation $\langle E_1E_2, h, E\rangle$ in which E_1E_2 is strongly connected.*

Proof. By Lemma 9.2, $\langle E_1E_2, h, E\rangle$ is a cascade simulation. From Lemma 9.3, Lemma 9.4, and Lemma 9.5, E_1E_2 is a strongly connected synchronizable cascade. ▮

10. MINIMALITY THEOREM (SIMULATION VERSION)

In this section we prove the minimality theorem, using theorems proved in Sections 6, 7.1, and 8.1. We then briefly examine its implications.

THEOREM 10.1 (Minimality Theorem—Simulation Version). *If E is a strongly connected automaton, $\langle E_1E_2, f, E\rangle$ is a clean synchronizable cascade simulation, and $\langle \dot{E}_1\dot{E}_2, h, E\rangle$ is a synchronizable cascade simulation constructed by our procedure, then there is a homomorphism g such that $\langle E_1E_2, g, \dot{E}_1\dot{E}_2\rangle$ is a clean synchronizable cascade simulation and $gh = f$.*

Suppose E is a strongly connected automaton and $\langle E_1E_2, f, E\rangle$ is a clean synchronizable cascade simulation. Let S, S_1, and S_2 be the state sets of E, E_1, and E_2 respectively. Let Q be the domain of f. So by the definition of simulation, Q is closed. Let $\bar{E}$ be the automaton formed from E_1E_2 by throwing away the states not in Q. Since the simulation is clean, $\bar{E}$ is strongly connected. Its state set is Q. We note that $f\colon Q \to S$.

By Theorem 8.5, the minimal size ranges of $\bar{E}$ are exactly the non-empty sets of form $(\{s_1\} \times S_2) \cap Q$, for $s_1 \in S_1$. Then by Theorem 7.5, the minimal size ranges of E are the non-empty sets of form $f[(\{s_1\} \times S_2) \cap Q]$, for $s_1 \in S_1$. For any $s_1 \in S_1$, we define $\overline{s_1}$ to be the set $f[(\{s_1\} \times S_2) \cap Q]$. Using this notation, we see that $\{\overline{s_1} \mid s_1 \in S_1\} - \{\varnothing\}$ is the set of minimal size ranges of E.

Now suppose we use our construction procedure to construct a synchronizable cascade simulation $\langle \dot{E}_1\dot{E}_2, h, E\rangle$, with $\dot{S}_1$ and $\dot{S}_2$ the state sets of $\dot{E}_1$ and $\dot{E}_2$, respectively. Then $\dot{S}_1 = \{\overline{s_1} \mid s_1 \in S_1\} - \{\varnothing\}$. This defines the states of $\dot{E}_1$ in terms of the states of E_1. We now look at the transition function of $\dot{E}_1$.

Suppose $(\{s_1\} \times S_2) \cap Q$ is a minimal size range of $\bar{E}$ and suppose r is an input symbol. Since Q is closed, the image $r[(\{s_1\} \times S_2) \cap Q]$ is a subset of $(\{r * s_1\} \times S_2) \cap Q$, and this last set, as we can see from its form, is (by Theorem 8.5) another minimal size range of $\bar{E}$. But by Theorem 6.2, the image must itself be a minimal size range, so $r[(\{s_1\} \times S_2) \cap Q] = (\{r * s_1\} \times S_2) \cap Q$. Replacing each side of this equation with its image under f and then using Eq. (2) and the definition of $\overline{s_1}$ gives $r[\overline{s_1}] = \overline{r * s_1}$. This equation holds provided the set $(\{s_1\} \times S_2) \cap Q$ was a minimal size range of $\bar{E}$. This proviso is equivalent to saying $(\{s_1\} \times S_2) \cap Q$ is non-empty, or to saying $\overline{s_1}$ is non-empty, or to saying $\overline{s_1} \in \dot{S}_1$. By Eq. (4), the transition function in $\dot{E}_1$ is given by $r * \dot{s}_1 = r[\dot{s}_1]$, for any $\dot{s}_1 \in \dot{S}_1$. So we now see that for $\overline{s_1} \in \dot{S}_1$,

$$r * \overline{s_1} = \overline{r * s_1}, \tag{10}$$

where the $*$ on the left is the transition function of $\dot{E}_1$, and the $*$ on the right is the transition function of E_1. This defines the transition function of $\dot{E}_1$ in terms of the transition function of E_1.

Remember that our construction of $\langle \dot{E}_1\dot{E}_2, h, E\rangle$ used an arbitrary set of correspondences

$$\varphi_{\dot{s}_1}\colon \dot{s}_1 \xrightarrow[onto]{1-1} \dot{S}_2. \tag{11}$$

Given h, we can figure out exactly what the correspondences were from Eq. (6), that is, from

$$h(\dot{s}_1, \dot{s}_2) = \varphi_{\dot{s}_1}^{-1}(\dot{s}_2). \tag{12}$$

The transition functions of $\dot{E}_2$ and of $\dot{E}_1\dot{E}_2$ are then given by Eqs. (7) and (8), that is, by

$$\langle r, \dot{s}_1 \rangle * \dot{s}_2 = \varphi_{r * \dot{s}_1}(r * h(\dot{s}_1, \dot{s}_2)), \tag{13}$$

$$r * \langle \dot{s}_1, \dot{s}_2 \rangle = \langle r * \dot{s}_1, \langle r, \dot{s}_1 \rangle * \dot{s}_2 \rangle. \tag{14}$$

Suppose $\langle s_1, s_2 \rangle \in Q$. Then the definition of $\overline{s_1}$ tells us that $f(s_1, s_2) \in \overline{s_1}$, so $\overline{s_1}$ is non-empty, and $\overline{s_1} \in \dot{S}_1$. Also, $f(s_1, s_2) \in \overline{s_1}$ and $\varphi_{\overline{s_1}}: \overline{s_1} \to \dot{S}_2$, so

$$\varphi_{\overline{s_1}}(f(s_1, s_2)) \in \dot{S}_2. \tag{15}$$

Then from Eq. (12), we have $h(\overline{s_1}, \varphi_{\overline{s_1}}(f(s_1, s_2))) = \varphi_{\overline{s_1}}^{-1}(\varphi_{\overline{s_1}}(f(s_1, s_2)))$, that is,

$$h(\overline{s_1}, \varphi_{\overline{s_1}}(f(s_1, s_2))) = f(s_1, s_2). \tag{16}$$

We now define $g: Q \to \dot{S}_1 \times \dot{S}_2$ by

$$g(s_1, s_2) = \langle \overline{s_1}, \varphi_{\overline{s_1}}(f(s_1, s_2)) \rangle. \tag{17}$$

From Eqs. (17) and (16), we have $h(g(s_1, s_2)) = f(s_1, s_2)$, so $gh = f$.

LEMMA 10.2. *g is a homomorphism.*

Proof. By Eq. (17), we have

$$g(r * s_1, \langle r, s_1 \rangle * s_2) = \langle \overline{r * s_1}, \varphi_{\overline{r * s_1}}(f(r * s_1, \langle r, s_1 \rangle * s_2)) \rangle.$$

Using $r * \langle s_1, s_2 \rangle = \langle r * s_1, \langle r, s_1 \rangle * s_2 \rangle$ makes this

$$g(r * \langle s_1, s_2 \rangle) = \langle \overline{r * s_1}, \varphi_{\overline{r * s_1}}(f(r * \langle s_1, s_2 \rangle)) \rangle. \tag{18}$$

From (15) and Eq. (13) we have

$$\langle r, \overline{s_1} \rangle * \varphi_{\overline{s_1}}(f(s_1, s_2)) = \varphi_{r * \overline{s_1}}(r * h(\overline{s_1}, \varphi_{\overline{s_1}}(f(s_1, s_2)))). \tag{19}$$

From this and Eq. (16), we obtain

$$\langle r * \overline{s_1}, \langle r, \overline{s_1} \rangle * \varphi_{\overline{s_1}}(f(s_1, s_2)) \rangle = \langle r * \overline{s_1}, \varphi_{r * \overline{s_1}}(r * f(s_1, s_2)) \rangle. \tag{20}$$

Then using Eq. (14) on the left and the fact that f is a homomorphism on the right gives

$$r * \langle \overline{s_1}, \varphi_{\overline{s_1}}(f(s_1, s_2)) \rangle = \langle r * \overline{s_1}, \varphi_{r * \overline{s_1}}(f(r * \langle s_1, s_2 \rangle)) \rangle.$$

Using Eq. (17) on the left and (10) and (18) on the right gives $r * g(s_1, s_2) = g(r * \langle s_1, s_2 \rangle)$. ▮

Proof. Proof of Theorem 10.1. Let the state sets of E_1E_2, $\dot{E}_1\dot{E}_2$, and E be $S_1 \times S_2$, $\dot{S}_1 \times \dot{S}_2$, and S, respectively. Let $Q = Dom(f)$. We are given a homomorphism $f: Q \to S$. We have seen how to write this as a composition of homomorphisms $g: Q \to \dot{S}_1 \times \dot{S}_2$ and $h: \dot{S}_1 \times \dot{S}_2 \to S$. The set $Ran(g)$ is closed by Theorem 7.1, and $\dot{S}_1 \times \dot{S}_2$ is strongly connected by Theorem 9.6, so $Ran(g) = \dot{S}_1 \times \dot{S}_2$. Q is closed and strongly connected since $\langle E_1E_2, f, E \rangle$ is a clean simulation. ▮

Let us examine the conclusion of Theorem 10.1 to see what it implies. Since $\langle E_1E_2, g, \dot{E}_1\dot{E}_2 \rangle$ is a clean synchronizable cascade simulation, and $\dot{E}_1\dot{E}_2$ is a strongly connected synchronizable cascade (by Theorem 9.6), we see by Theorem 8.7 that g collapses onto some function g_1, and $\langle E_1, g_1, \dot{E}_1 \rangle$ is a clean simulation. In fact, Theorem 7.8 tells us that g_1 is the unique function for which $\langle E_1, g_1, \dot{E}_1 \rangle$ is a clean simulation.

Thus E_1E_2 simulates $\dot{E}_1\dot{E}_2$ in a very structured way, in which E_1 simulates $\dot{E}_1$, and then E_2 does whatever else is necessary to make the simulation deal with $\dot{E}_2$ as well.

11. MINIMALITY THEOREM (DECOMPOSITION VERSION)

In this section we define "decomposition" in terms of "simulation." We translate the minimality theorem into a theorem stated in terms of decompositions. We then discuss its implications.

We say that a cascade E_1E_2 is a **synchronizable cascade decomposition** of an automaton E if there is a homomorphism h such that $\langle E_1E_2, h, E \rangle$ is a synchronizable cascade simulation. We are, of course, interested in cascade decompositions of a *strongly connected* finite automaton E. Theorem 9.6 tells us that our construction procedure constructs a strongly connected synchronizable cascade decomposition of E. Now what about other synchronizable cascade decompositions of E?

THEOREM 11.1 (Minimality Theorem—Decomposition Version). *If E_1E_2 is a synchronizable cascade decomposition of a strongly connected automaton E, then any decomposition of E constructed by our procedure is a homomorphic image of a strongly connected subautomaton of E_1E_2. Consequently, either E_1E_2 has more states than the constructed decompositions of E have, or E_1E_2 is isomorphic to a constructed decomposition of E.*

Proof. Suppose E_1E_2 is a synchronizable cascade decomposition of a strongly connected automaton E, and $\dot{E}_1\dot{E}_2$ is a constructed decomposition of E. Then by Theorem 7.7, there is a clean synchronizable cascade simulation $\langle E_1E_2, f, E \rangle$, and consequently by Theorem 10.1 there is a clean synchronizable cascade simulation $\langle E_1E_2, g, \dot{E}_1\dot{E}_2 \rangle$. $Dom(g)$ is a closed strongly connected set of states, so this set is the state set of a strongly connected subautomaton of E_1E_2. ▮

We see from Theorem 11.1 that the constructed decompositions of E are minimal synchronizable cascade decompositions of E, and that all minimal synchronizable

cascade decompositions of E are isomorphic. In particular, all constructed decompositions of E are isomorphic. Note then that since the constructed decompositions are strongly connected, so are all the minimal synchronizable cascade decompositions.

But we must be careful not to jump to conclusions that are too strong. Just because two synchronizable cascades are isomorphic doesn't mean there are no differences at all between them. Of course if they are isomorphic, they are very much alike. For example, the following theorem shows that if they are strongly connected, then their first components are isomorphic.

THEOREM 11.2. *If E_1E_2 and $\dot{E}_1\dot{E}_2$ are isomorphic strongly connected synchronizable cascades, then E_1 and $\dot{E}_1$ are isomorphic.*

Proof. Let $S_1 \times S_2$ and $\dot{S}_1 \times \dot{S}_2$ be the state sets of E_1E_2 and $\dot{E}_1\dot{E}_2$, respectively. Suppose $h: S_1 \times S_2 \to \dot{S}_1 \times \dot{S}_2$ is an isomorphism onto. Then $\langle E_1E_2, h, \dot{E}_1\dot{E}_2\rangle$ and $\langle \dot{E}_1\dot{E}_2, h^{-1}, E_1E_2\rangle$ are both clean synchronizable cascade simulations. Then Theorem 8.7 gives us two clean simulations, $\langle E_1, f, \dot{E}_1\rangle$ and $\langle \dot{E}_1, g, E_1\rangle$. So S_1 and $\dot{S}_1$ are the same size, and f and g are isomorphisms between E_1 and $\dot{E}_1$. ∎

So all minimal synchronizable cascade decompositions of E have isomorphic first components. But their second components can of course be very different.

Which second component our construction procedure constructs depends on which set of correspondences the procedure uses. What about second components of other minimal synchronizable cascade decompositions of E? Is each of these isomorphic to one of the second components our construction procedure can construct? In a sense the answer is "yes," but in this paper's definition of "homomorphism," and hence of "isomorphism," we insist that the two automata being compared have the same input alphabet. The states of the first component are inputs to the second component. We shall see that if we re-name first component states appropriately, then we can answer our question in the affirmative.

Given a strongly connected automaton E, every decomposition constructed by our procedure has the same first component E_1. We shall call this the *canonical first component* of E. Its state set S_1 is the set of minimal size ranges of E, and its transition function is given by Eq. (4). Since all minimal synchronizable cascade decompositions of E are isomorphic, Theorem 11.2 tells us that their first components are all isomorphic to the canonical first component. So if we have a minimal synchronizable cascade decomposition of E, we can simply re-name the states of the first component to convert the first component into the canonical first component, and the result will be a minimal synchronizable cascade decomposition E_1E_2 in which E_1 is the canonical first component. Of course the new "name" of a first component state is a minimal size range of E. We take a look at the decomposition E_1E_2 after re-naming. We shall show that it is identical to one of our constructed decompositions.

THEOREM 11.3. *If E_1E_2 is a minimal synchronizable cascade decomposition of a strongly connected automaton E, and if E_1 is the canonical first component of E, then E_1E_2 is identical to one of our constructed decompositions of E.*

Proof. By definition, we have a homomorphism h such that $\langle E_1E_2, h, E\rangle$ is a simulation. We let S and $S_1 \times S_2$ be the state sets of E and E_1E_2, respectively. We need only show that there are correspondences $\varphi_{s_1}: s_1 \xrightarrow[onto]{1-1} S_2$ (one correspondence for each $s_1 \in S_1$) such that Eqs. (5)–(8) hold.

Since E_1E_2 is a cascade, Eq. (8) holds. (I.e., Eq. (3) holds.) Since E_1 is the canonical first component, Eq. (4) holds. And of course Eq. (9), the extension of Eq. (4), also holds. Since E_1E_2 is minimal, it is strongly connected. Since $Dom(h)$ is closed, $Dom(h) = S_1 \times S_2$. Suppose s_1 and $\hat{s}_1$ are two (not necessarily different) states in S_1. Now s_1 is a minimal size range of E. Let ρ be a string such that $s_1 = Ran(\rho)$. By Eq. (9), the E_1 state $\rho * \hat{s}_1$ is the image $\rho[\hat{s}_1]$ in E. By Theorem 6.1, this image is $Ran(\rho)$, which is s_1. Thus in E_1 we have $\rho * \hat{s}_1 = s_1$.

By Corollary 8.6, the set $\{\hat{s}_1\} \times S_2$ is a minimal size range of E_1E_2. Now $\rho[\{\hat{s}_1\} \times S_2] \subseteq \{\rho * \hat{s}_1\} \times S_2$. Theorem 6.2 tells us that the left side is a minimal size range of E_1E_2, and Corollary 8.6 tells us that the right side is a minimal size range of E_1E_2. Thus the two sides are equal. So, using $\rho * \hat{s}_1 = s_1$, we have $\rho[\{\hat{s}_1\} \times S_2] = \{s_1\} \times S_2$. Applying h to both sides and using Eq. (2) gives $\rho[h[\{\hat{s}_1\} \times S_2]] = h[\{s_1\} \times S_2]$. By Theorem 7.5, $h[\{\hat{s}_1\} \times S_2]$ is a minimal size range of E. Then by Theorem 6.1, $\rho[h[\{\hat{s}_1\} \times S_2]] = Ran(\rho) = s_1$. So from the last two equations, we have $h[\{s_1\} \times S_2] = s_1$. The s_1 was arbitrary, so this holds for all s_1 in S_1.

Consider any constructed decomposition of E. It has the same number of states as does E_1E_2 (since it is isomorphic to E_1E_2) and its first component is also E_1, so its second component has the same number of states as does E_2. Now by our construction procedure, its second component has the same number of states as does every minimal size range of E. Therefore, S_2 has the same number of states as does every minimal size range of E.

For each s_1 in S_1, we define $\vartheta_{s_1}: S_2 \to S$ by $\vartheta_{s_1}(s_2) = h(s_1, s_2)$. We have seen that $h[\{s_1\} \times S_2] = s_1$, so $Ran(\vartheta_{s_1}) = s_1$. Now s_1 is a minimal size range of E and so we have just seen that it has the same number of states as S_2. Therefore, $\vartheta_{s_1}: S_2 \xrightarrow[onto]{1-1} s_1$. We define $\varphi_{s_1} = \vartheta_{s_1}^{-1}$, for each s_1 in S_1. So $\varphi_{s_1}: s_1 \xrightarrow[onto]{1-1} S_2$, and $h(s_1, s_2) = \varphi_{s_1}^{-1}(s_2)$. So Eqs. (5) and (6) hold. By Eq. (8), we have $h(r * \langle s_1, s_2\rangle) = h(r * s_1, \langle r, s_1\rangle * s_2)$. If we use the previous equation on the right and use the fact that h is a homomorphism on the left, this equation becomes $r * h(s_1, s_2) = \varphi_{r*s_1}^{-1}(\langle r, s_1\rangle * s_2)$. Applying φ_{r*s_1} to both sides gives $\varphi_{r*s_1}(r * h(s_1, s_2)) = \langle r, s_1\rangle * s_2$. This is Eq. (7). So Eqs. (5)–(8) hold, and so E_1E_2 is identical to one of our constructed decompositions. ▮

12. CONCLUSION

There are many approaches to constructing a synchronizable cascade decomposition of a strongly connected finite automaton. One approach was given in [8]. Another approach is given in this paper. Both these approaches yield a minimal synchronizable cascade decomposition. We have proved that, up to isomorphism, there is really only one minimal synchronizable cascade decomposition, and that furthermore this decomposition is a homomorphic image of all the other

synchronizable cascade decompositions (or rather of subautomata of them). This gives us a feeling for how the different synchronizable cascade decompositions are related.

We have pointed out that, although up to isomorphism there is only one minimal synchronizable cascade decomposition, this does not mean that all minimal synchronizable cascade decompositions have isomorphic second components. It does, however, mean that they have isomorphic first components. We said in the Introduction that in our applications we are interested mainly in the first components of the synchronizable cascade decompositions, regarding the second components mainly as merely sources of noise. Thus we are happy to note that all minimal synchronizable cascade decompositions of a strongly connected automaton E have isomorphic first components, and that we can obtain such a first component if we construct a decomposition of E in the way given here.

The description we gave here of the first component in terms of minimal size ranges is particularly simple and we have found it easy to use. It is comforting to know that any other description of the first component of a minimal synchronizable cascade decomposition is describing the same automaton (up to isomorphism).

This paper provides a solid basis for the use of synchronizable cascade decompositions in the analysis of reward schemes for learning systems. Such analysis frequently involves constructing such a decomposition and then introducing probability distributions over input symbols and states. The simpler the decomposition construction method, the simpler the subsequent analysis is likely to be. But a method conceptually simple in one context may not be simple in another. With the knowledge we now have of how the different synchronizable cascade decompositions are related, we can more easily choose the method appropriate to the analysis.

REFERENCES

1. M. A. Arbib, "Theories of Abstract Automata," Prentice–Hall, Engelwood Cliffs, NJ, 1969.
2. J. J. Grefenstette, Credit assignment in rule discovery systems based on genetic algorithms, *Mach. Learning* **3** (1988), 225–246.
3. J. H. Holland, Adaptive plans optimal for payoff-only environments, *in* "Proceedings of the Second Hawaii Conference on System Sciences," pp. 917–920, University of Hawaii, Honolulu, 1969.
4. J. H. Holland, "Adaptation in Natural and Artificial Systems," Univ. of Michigan Press, Ann Arbor, 1975.
5. J. H. Holland, Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule based systems, *in* "Machine Learning, II" (R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds.), pp. 593–623, Morgan Kaufmann, Los Altos, CA, 1986.
6. J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard, "Induction: Processes of Inference, Learning and Discovery," MIT Press, Cambridge, MA, 1986.
7. T. H. Westerdale, A reward scheme for production systems with overlapping conflict sets, *IEEE Trans. Systems Man Cybernet.* **16** (1986), 369–383.
8. T. H. Westerdale, An automaton decomposition for learning system environments, *Inform. and Comput.* **77** (1988), 179–191.
9. T. H. Westerdale, Quasimorphisms or queasymorphisms? Modeling finite automaton environments, *in* "Foundations of Genetic Algorithms" (G. J. E. Rawlins, Ed.), pp. 128–147, Morgan Kaufmann, San Mateo, CA, 1991.

10. T. H. Westerdale, Classifier systems—No wonder they don't work, *in* "Proceedings of the Second Annual Conference on Genetic Programming" (J. R. Koza *et al.*, Eds.), pp. 529–537, Morgan Kaufmann, San Francisco, CA, 1997.
11. T. H. Westerdale, An approach to credit assignment in classifier systems, *Complexity* **4** (1999), 49–52; for the full version see http://journals.wiley.com/complexity.
12. T. H. Westerdale, Local reinforcement and recombination in classifier systems, *Evolutionary Comput.* **9** (2001), 259–281.